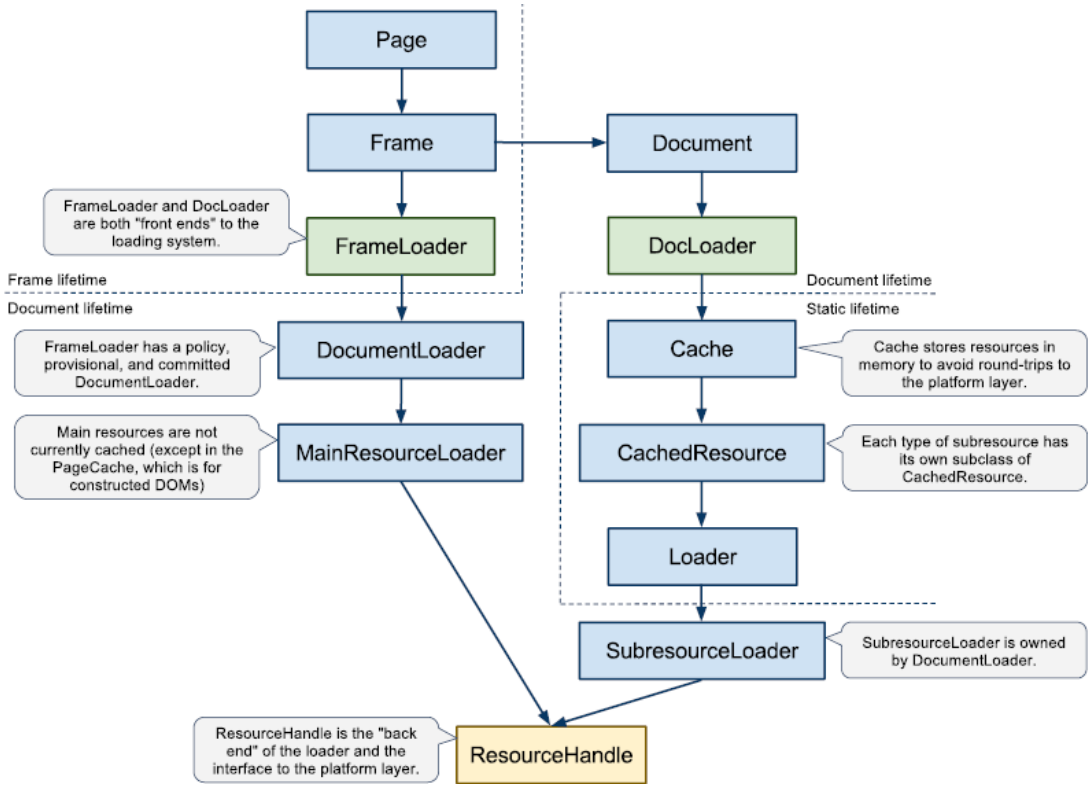
WebKit 加载网页的流程

dlmu2001

在 WebKit 渲染一个页面之前，它需要从网络上（其实也可以从本地文件或者内存加载）加载页面以及和它相关的所有派生资源。同加载资源相关的层有很多，在本文中，我将聚焦于解释 WebCore，这一 WebKit 的主要渲染模块，如何参与到加载过程中的。

WebKit 有两条加载路线，一条是加载 documents 到 frames 里面，另一条是加载派生资源（比如图片和脚本）。下图总结出了这两条路线涉及到的主要对象。



加载 Frames

FrameLoader 类负责将 documents 加载到 Frames。当你点击一个链接的时候，FrameLoader 创建一个新的处于”policy” 状态的 DocumentLoader 对象，等待 WebKit 客户端决定是否处理这个加载。通常 WebKit 客户端会指示 FrameLoader 将这个加载视为一个导航（navigation），而不是阻止加载等。

一旦客户端指示 FrameLoader 将本次加载视为一个导航，FrameLoader 就推动 DocumentLoader 进入”provisional”状态，在该状态，DocumentLoader 会发起一个网络请求，并等待以确定网络请求将发起一个下载还是一个新的 document。

接下去，DocumentLoader 会创建一个 MainResourceLoader 对象，这个对象主要用来通过 ResourceHandle 接口同平台网络库进行交互。将 MainResourceLoader 和 DocumentLoader 分开来主要有两个目的：（1）MainResourceLoader 让 DocumentLoader 从处理 ResourceHandle 回调的细节中抽身出来（2）降低 MainResourceLoader 的生命周期和 DocumentLoader 的生命周期（同 Document 绑定）的耦合度。

一旦加载系统接收到足够的信息可以确定资源确实代表了 document，FrameLoader 就将 DocumentLoader 推向”committed”状态，在该状态中，frame 将显示 document。

加载派生资源

显示一个 Web 页面，不只是需要组成 document 的 HTML，还需

要加载 document 引用到的图片，脚本及其它派生资源。DocLoader 类负责加载这些派生资源。(注意 DocumentLoader 和 DocLoader 有非常相似的名字，但是他们的作用是完全不同的)。

以加载一个图片为例，加载图片时，DocLoader 首先询问 Cache，在内存中是否有该图片的拷贝（即 CachedImage 对象)。如果图片已经在 Cache 中，DocLoader 可以以该 Cache 中的图片立即响应。更有效率更高者，Cache 在 video 内存保留了解码过的图片，这样 webkit 连解码图片的过程都不需要了。

如果图片不再 Cache 中，Cache 就会创建一个新的 CachedImage 对象来代表 image。然后 CachedImage 对象要求 Loader 对象发起一个网络请求，Loader 对象创建 SubresourceLoader。SubresourceLoader 在派生资源加载路线上的作用同 MainResourceLoader 在主资源加载路线上的作用一样，它也是用来同 ResourceHandle 直接打交道。

可以优化的地方

加载路线上有很多地方可以优化。FrameLoader 过于复杂，承担了不止是加载一个 frame 的任务。比如，FrameLoader 有好几个非常相近的名为“load”的方法，非常容易混淆，并负责创建新窗口，这个同加载 frame 是没有关系的。另外，有很多越级行为，比如，MainResourceLoader 将收到的字节直接传给了 FrameLoader，而不是 DocumentLoader。

从上图可以看出，Cache 只要于派生资源（subresource)。主资源

的加载并没有从 WebKit 的内存 cache 中获益。如果我们结合这两条路线，我们可以提高主资源加载的性能。